(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0070625
(43) 공개일자 2022년05월31일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2018.01) *G06F 9/455* (2018.01)
G06F 9/50 (2018.01)
(52) CPC특허분류
G06F 9/4881 (2013.01)
G06F 9/45558 (2013.01)
(21) 출원번호 10-2020-0157353
(22) 출원일자 2020년11월23일
심사청구일자 2020년11월23일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
이재학
경기도 성남시 분당구 돌마로 486번길 12, 216동 901호
유헌창
서울특별시 노원구 노원로 214, 6동 202호
(뒷면에 계속)
(74) 대리인
김홍석

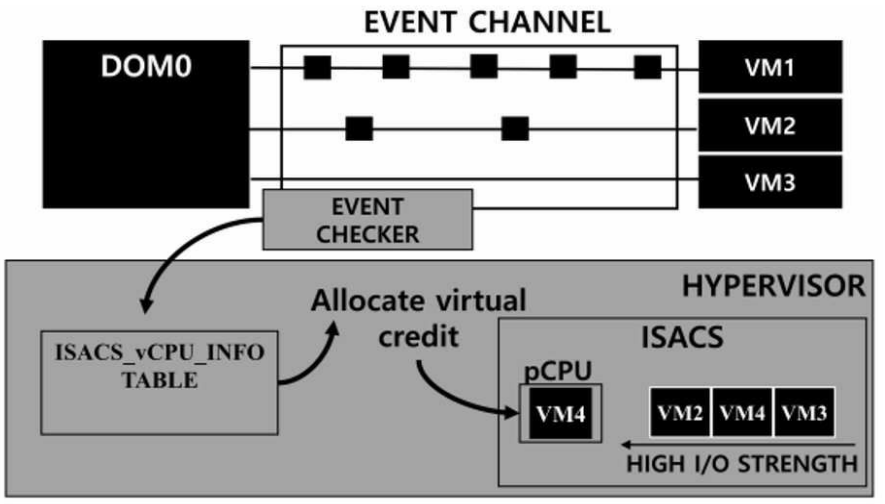
전체 청구항 수 : 총 6 항

(54) 발명의 명칭 **가상화 환경에서 가상머신의 I/O 성능을 높이기 위한 스케줄링 방법**

(57) 요 약

크레딧 스케줄링 방법을 개시한다. 크레딧 스케줄링 방법은 가상머신의 I/O 집약적 워크로드를 처리하는 vCPU에 대한 I/O 정보를 수집하고, 상기 수집된 I/O 정보를 통해 I/O 워크로드 정도에 따라 매핑된 I/O 가상 크레딧(I/O virtual credit: IVCredit)을 I/O 집약적 vCPU에 할당하며, 상기 가상머신의 CPU 작업의 지속성을 보장하기 위하여 CPU 집약적 vCPU에 CPU 가상 크레딧(CPU virtual credit: CVCredit)을 할당하는 것을 포함한다.

대 표 도 - 도3



(52) CPC특허분류
G06F 9/5027 (2013.01)
G06F 9/5077 (2013.01)
G06F 2009/45579 (2019.08)
(72) 발명자
정진원
서울특별시 중구 다산로 32, 42동 601호

장용현
서울특별시 성북구 고려대호22길 45, 101호

이 발명을 지원한 국가연구개발사업

과제고유번호	1711103248
과제번호	2018-0-00480-003
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	SW컴퓨팅산업원천기술개발(R&D)
연구과제명	커넥티드카 모빌리티 실시간 서비스를 위한 엣지 클라우드 플랫폼개발
기 여 율	1/1
과제수행기관명	(주)에이티지랩
연구기간	2020.01.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 크레딧 스케줄링 방법에 있어서,

가상머신의 I/O 집약적 워크로드를 처리하는 vCPU에 대한 I/O 정보를 수집하는 단계;

상기 수집된 I/O 정보를 통해 I/O 워크로드 정도에 따라 매핑된 I/O 가상 크레딧(I/O virtual credit: IVCredit)을 I/O 집약적 vCPU에 할당하는 단계; 및

상기 가상머신의 CPU 작업의 지속성을 보장하기 위하여 CPU 집약적 vCPU에 CPU 가상 크레딧(CPU virtual credit: CVCredit)을 할당하는 단계를 포함하는 크레딧 스케줄링 방법.

청구항 2

제1항에 있어서,

상기 가상머신의 I/O 집약적 워크로드를 처리하는 vCPU에 대한 I/O 정보를 수집하는 단계는,

드라이브 도메인에서 특정 vCPU로 전송되는 I/O 이벤트를 이벤트 채널에 배치된 이벤트 체커를 통해 확인하는 단계;

상기 이벤트 체커에서 상기 드라이브 도메인의 이벤트 채널에 바인딩된 가상머신의 특정 이벤트 채널을 추적하는 단계;

상기 이벤트 체커에서 상기 드라이브 도메인이 상기 가상머신으로 전송하는 I/O 이벤트 수를 측정하는 단계;

특정 런-큐에서 크레딧 재할당(credit reallocation)이 호출되면, 상기 런-큐에 있는 모든 vCPU에 대한 I/O 이벤트 정보를 가져 오도록 상기 이벤트 체커에 신호를 송신하는 단계; 및

상기 이벤트 체커에서 상기 신호를 수신하면, 상기 특정 런-큐의 vCPU 별 I /O 이벤트 정보를 전송하고 모든 vCPU의 I/O 이벤트 정보를 초기화하는 단계를 포함하는 크레딧 스케줄링 방법.

청구항 3

제1항에 있어서,

상기 수집된 I/O 정보를 통해 I/O 워크로드 정도에 따라 매핑된 I/O 가상 크레딧(IVCredit)을 I/O 집약적 vCPU에 할당하는 단계는,

드라이브 도메인에서 특정 vCPU에 대하여 모든 I/O 이벤트에서 평균 I/O 이벤트 간격($\overline{I_e}$)을 하기의 수학식 1을 통해 계산하는 단계;

상기 계산된 $\overline{I_e}$ 을 기반으로 상기 가상머신의 각 vCPU에 대한 상기 IVCredit을 하기의 수학식 2를 통해 계산하는 단계; 및

특정 런-큐에서 크레딧 재할당(credit reallocation)이 호출될 때마다, 상기 IVCredit를 상기 I/O 집약적 vCPU에 할당하는 단계를 포함하는 크레딧 스케줄링 방법.

[수학식 1]

$$\overline{I_e} = 0.9 \times I_e + (1 - 0.9) \times \overline{I_{e-1}}, \quad where \;\; t_e - t_{e-1} \;\; is \;\; I_e$$

여기서, $t_e - t_{e-1}$ 는 상기 드라이버 도메인이 특정 가상머신의 I/O 요청에 대한 I/O 응답을 수행하기 위해 발

생시키는 I/O 응답 이벤트에 대한 간격을 표현하며, I_e 로 표현될 수 있음. t_{e-1}는 이전 I/O 응답 이벤트 발생한 시간, t_e 는 현재 발생된 I/O 응답 이벤트가 발생한 시간을 의미함.

[수학식 2]

$$IVCredit = E_n \times \overline{I_e} \times \frac{consumed\ credit}{\frac{\sum_i^n cc_i}{n}}$$

여기서, $E_n \times \overline{I_e}$ 는 vCPU에 대해 드라이버 도메인으로부터 수신한 I/O 응답 이벤트 수와 EMA가 적용된 I/O 이벤트의 평균 간격을 곱하는 것을 표현한 것이며, 이는 런-큐 내 vCPU 중에서 특정 vCPU의 차별화된 I/O 강도를 도출하는 과정을 표현함. 런-큐 내부에서 크레딧 재할당이 호출될 되는 상황에 대하여, consumed credit은 런-큐에서 크레딧 재할당이 발생하기까지의 실질적인 vCPU가 pCPU를 점유하여 소비한 크레딧의 양, $\frac{\sum_i^n cc_i}{n}$ 은 런-큐 내부의 전체 vCPU들의 평균 크레딧의 양을 표현함. 크레딧 재할당이 발생할 때, E_n 은 드라이버 도메인이 특정 가상머신의 vCPU로 I/O 응답 이벤트를 발생시킨 횟수를 의미함.

청구항 4

제3항에 있어서,

상기 CPU 집약적 vCPU에 CVCredit을 할당하는 단계는,

하기 수학식 3을 이용한 CVCredit을 할당받을 대상을 정하는 단계로써, CPU-부스트 그룹을 형성하는 단계; 및

하기 수학식 3을 통해 도출해낸 CPU-부스트 그룹에 속하는 vCPU들에게 수학식 4를 통해 구한 CVCredit을 제공하는 단계; 및

상기 특정 vCPU가 크레딧 재할당(credit reallocation)을 호출하여 CPU-부스트 상태가 되어 CPU-부스트 그룹에 있는 vCPU만을 스케줄링하는 단계; 및

상기 CVCredit는 상기 런-큐에 있는 모든 vCPU에게 할당된 IVCredit의 양을 기반으로 할당되며 상기 크레딧 재할당(credit reallocation)을 호출한 vCPU의 크레딧이 0일 때 추가로 할당된 CVCredt을 소비하는 단계로써, CPU-부스트 상태를 형성하는 단계를 포함하는 크레딧 스케줄링 방법.

[수학식 3]

$$\frac{\sum_{i=1}^n IVCredit_i}{k} \times \frac{1}{IVCredit_c + 1} > 1$$

[수학식 4]

$$CVCredit = \frac{\sum_{i=1}^n IVCredit_i}{j} \times \frac{consumed\ credit}{default\ credit}$$

청구항 5

제1항에 있어서,

상기의 수학식 1 내지 수학식 4 및 하기 알고리즘 1을 기반으로, 각 런-큐의 I/O 워크로드 정도에 따른 IVCredit 할당 및 CPU 성능 저하를 방지하기 위한 CVCredit 할당을 수행하는 크레딧 스케줄링 단계; 및

크레딧 재할당을 수행하는 단계를 더 포함하되,

상기 크레딧 재할당은 CVCredit을 할당받은 vCPU를 제외한 모든 vCPU은 같은 양의 IVCredit을 할당받는 크레딧 스케줄링 방법.

[알고리즘 1]

Algorithm 1 ISACS: Scheduling algorithm

$vCPU_c$: Current vCPU on the pCPU

$vCPU_n$: Next vCPU to Run

```
if CPU − BOOST STATE == FALSE then
    Burn the vCPU_c.credit
    Set vCPU with highest credits for all vCPUs in the RunQ as vCPU_n
    if vCPU_n.credit ≦ vCPU_n.endcredit then
        CPU − BOOST STATE ← TRUE
        Set vCPU with highest CVCredits over all vCPUs in CPU − BOOST GROUP as vCPU_c
    end if
end if
if CPU − BOOST STATE == TRUE then
    Burn the vCPU_c.CVCredit
    Set vCPU with highest CVCredits over all vCPUs in CPU − BOOST GROUP as vCPU_n
    if vCPU_n.credit ≦ vCPU_i.credit then
        RunQ.IOload ← 0
        Init the member of CPU − BOOST GROUP
        CPU − BOOST STATE ← FALSE
        for vCPU_i ∈ RunQ do
            Derive IVCredit for vCPU_i using Equation 1 and Equation 2
            vCPU_i.credit ← default credit + IVCredit
            vCPU_i.endcredit ← IVCredit
            RunQ.IOload+ = IVCredit
        end for
        for vCPU_i ∈ RunQ do
            if Equation 3 is satisfied then
                Join vCPU_i to the CPU − BOOST GROUP
                vCPU_i.CVCredit ← CVCredit which is calculated by using Equation 4
            end if
        end for
    end if
end if
return vCPU_n
```

청구항 6

제5항에 있어서,

상기 크레딧 재할당을 수행한 후,

하기의 수학식 5, 수학식 6, 및 알고리즘 2를 기반으로, 각 런-큐의 I/O 워크로드 정도에 따라 vCPU의 마이그레이션을 수행하는 단계를 더 포함하는 크레딧 스케줄링 방법.

[수학식 5]

$$RunQ_i.IOevent < RunQ_s.IOevent$$

[수학식 6]

$$RunQ_d.IOload + vCPU_j.IVCredit < RunQ_s.IOload - vCPU_j.IVCredit$$

[알고리즘 2]

Algorithm 2 ISACS: I/O load-balancing

```
for RunQ_i ∈ RunQ do
  if RunQ_i.id == RunQ_s.id then
    continue
  end if
  if RunQ_i.IOevent < RunQ_s.IOevent then
    RunQ_d ← RunQ_i
  end if
end for
for vCPU_j ∈ RunQ_s do
  if RunQ_d.IOload + vCPU_j.IVCredit < RunQ_s.IOload − vCPU_j.IVCredit then
    vCPU_m ← vCPU_j
  end if
end for
if vCPU_m!=NULL then
  Migrate vCPU_m to RunQ_d
  Return
end if
Return
```

발명의 설명

기 술 분 야

[0001] 본 발명은 가상화 환경에서 발생하는 입출력 작업에 대한 성능 하락을 완화시키고 성능을 높이기 위한 가상화 서버 시스템의 자원 관리 기법에 관한 것이다.

배 경 기 술

[0002] 이하의 기술 내용은 단순히 본 발명의 실시예에 대한 배경 정보를 제공할 뿐 종래 기술을 구성하는 것은 아니다.

[0003] 가상화 환경의 서버는 호스트 운영체제 없이 하드웨어들이 가상화 되어 소프트웨어 계층인 하이퍼바이저(VMM)에 의해 관리되어 드라이버 도메인 및 게스트 도메인들의 OS들은 가상화 환경에 적합하도록 수정된 OS 커널을 가지고 수행하는 서버 환경이거나, 하드웨어를 관리하는 호스트 운영체제가 존재하고, 그 호스트 운영체제에 하이퍼바이저가 동작하는 서버 환경을 포함한다.

[0004] 가상화 환경에서, 게스트 도메인은 사용자들에게 제공되는 가상머신을 뜻하고, 드라이버 도메인은 게스트 도메인의 I/O들을 실질적으로 처리하는 하드웨어 드라이버 집합 또는 장치 드라이버 접근을 위한 QEMU를 가지고 있는 가상머신을 뜻한다.

[0005] 앞서 언급한 게스트 도메인의 I/O에 대해서, 모든 게스트 도메인들의 I/O들은 드라이버 도메인으로 프론트엔드 드라이버를 통해 I/O 요청을 보내고, 드라이버 도메인은 백엔드 드라이버를 통해 수신 후 실질적인 I/O 요청을 처리한다. 드라이버 도메인은 해당 I/O 요청를 처리한 후, 해당 I/O 요청에 대한 I/O 응답을 게스트 도메인에게 전송하는 분리 드라이버 모델을 가진 가상화 환경을 뜻한다.

[0006] 가상화 환경에서, 모든 도메인들은 논리적인 CPU인, 가상 씨피유(vCPU)를 가지고 있다. 일반적인 OS 환경과는 달리 가상화 환경에서는, 하이퍼바이저 스케줄러(VMM 스케줄러, VM 스케줄러, 가상머신 스케줄러)에 의해 vCPU들이 물리 씨피유(pCPU)를 점유해야 도메인의 실질적인 작업(화면 출력, I/O 작업, 연산작업 등)이 수행된다.

[0007] 이에 대하여, 모든 도메인들은 하이퍼바이저 스케줄러 정책에 의해 자원 점유 정도가 결정된다. 기존 가상화 환경은 도메인들 간의 공정한 자원 점유 정도를 제공하기 위해 도메인의 vCPU가 pCPU를 점유하는 시간 또는 정도를 기준으로 vCPU들이 스케줄링 되었다. 하지만, pCPU 자원 점유 정도를 기준으로 vCPU들을 스케줄링하는 하이퍼바이저 스케줄러에 의해 I/O-집약적인 가상머신들의 I/O 성능은 기존 비 가상화 환경(베어메탈 환경)에서의 성능보다 낮은 성능을 가지고 있다. pCPU 자원 점유 정도를 기준으로 vCPU들을 스케줄링할 시, 가상화 환경에서 I/O 성능이 낮은 이유는 기존 가상화 환경에서는 vCPU가 처리하고 있는 작업 유형 특성 및 가상머신의 I/O 발생 정도를 고려하지 않을 뿐더러, 가상머신에서 발생하는 I/O의 불예측성에 대한 I/O-집약적인 vCPU의 pCPU 점유 보장에 대한 보장성이 낮기 때문이다. 이는 곧, 클라우드 서버 또는 가상화된 시스템에서 I/O-집약적인 어플리케이션을 수행하는 클라우드 사용자 또는 가상머신의 I/O 성능을 보장하지 못하며 결국에는 사용자에 대한

QoS(Quality of Service)를 떨어뜨리는 경우가 발생 된다. 그러한 이유로, 가상화 환경에서 I/O 성능을 높이기 위해서는 vCPU의 작업 유형을 인식하고, I/O 발생 정도를 고려하여 I/O-집약적인 pCPU 점유 보장을 통해 입출력 성능을 높이는 것이 중요하다. 또한, I/O-집약적인 작업을 수행하는 vCPU의 pCPU 점유 보장으로 인한 CPU-집약적인 vCPU의 CPU 작업 성능 저하를 방지하는 것도 중요하다. 마지막으로, I/O-집약적인 vCPU들이 특정 런-큐에 집중되는 것을 방지하기 위해 vCPU 마이그레이션을 통하여 I/O-집약적인 vCPU들이 런-큐들에 대하여 고르게 분산시킬 수 있도록 하는 것이 중요하다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) 대한민국 등록특허 제1707601호 (2017.02.16. 공고)

(특허문헌 0002) 대한민국 등록특허 제1743028호 (2017.06.02. 공고)

발명의 내용

해결하려는 과제

[0009] 본원 발명이 해결하고자 하는 과제는 클라우드 서버 또는 가상화된 시스템에서 기존 환경보다 높은 I/O 성능을 나타내는 크레딧 스케줄링 방법에 관한 것이다.

과제의 해결 수단

[0010] 해결하고자 하는 과제를 달성하기 위하여 본 발명의 실시예들에 따른 크레딧 스케줄링 방법은, 가상머신의 I/O 집약적 워크로드를 처리하는 vCPU에 대한 I/O 정보를 수집하는 단계, 상기 수집된 I/O 정보를 통해 I/O 워크로드 정도에 따라 매핑된 I/O 가상 크레딧(I/O virtual credit: IVCredit)을 I/O 집약적 vCPU에 할당하는 단계, 및 상기 가상머신의 CPU 작업의 지속성을 보장하기 위하여 CPU 집약적 vCPU에 CPU 가상 크레딧(CPU virtual credit: CVCredit)을 할당하는 단계를 포함한다.

발명의 효과

[0011] 본 발명의 실시예들에 따른 스케줄링 방법에 의할 경우, 기존 가상화 서버에서 가상머신에 할당된 vCPU의 pCPU 점유 정도에 따른 스케줄링 정책으로 가상머신의 낮은 I/O 성능을 높일 수 있는 효과가 있다.

[0012] 구체적으로, I/O-집약적인 작업을 수행하는 가상머신의 vCPU에 대한 pCPU 점유 보장을 통하여 입출력 성능을 향상시킬 수 있으며, I/O-집약적인 작업을 수행하는 가상머신의 vCPU의 pCPU 점유 보장으로 인한 CPU-집약적인 vCPU의 CPU 작업 수행을 보장하여 CPU-집약적인 가상머신의 CPU 처리 성능 또한 보장할 수 있다.

[0013] 마직막으로, I/O-집약적인 vCPU들이 특정 런-큐에 집중되는 것을 방지하기 위해 vCPU 마이그레이션을 통하여 I/O-집약적인 vCPU들이 런-큐들에 대하여 고르게 분산시킬 수 있다.

도면의 간단한 설명

[0014] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 종래의 가상화 환경에서 I/O 요청 및 응답에 대한 수행 과정을 설명하기 위한 모식도이다.

도 2는 본 발명의 일 실시예에 따른 크레딧 스케줄링 방법을 설명하기 위한 순서도이다.

도 3은 본 발명의 일 실시예에 따른 크레딧 스케줄러를 설명하기 위한 모식도이다.

도 4는 본 발명의 일 실시예에 따른 크레딧 스케줄러의 통합 vCPU 정보 구조체를 설명하기 위한 도면이다.

도 5는 본 발명의 일 실시예에 따른 크레딧 스케줄링 결과를 설명하기 위한 도면이다.

도 6은 도 5에 따른 런-큐에 대한 vCPU 배치 또는 스케줄링 순서를 나타내며, CPU-부스트 그룹에 존재하는 vCPU들의 CVCredit을 소비하기 위한 CPU-부스트 상태와, CPU-부스트 상태가 아닐 때 IVCredit을 소비하는 상황을 표현할 수 있다.

도 7은 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존 환경에 대하여 네트워크 입출력 집약적인 가상머신의 네트워크 I/O에 대한 지연시간을 나타낸다.

도 8은 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 입출력과 CPU 작업을 동시에 수행하는 가상머신의 네트워크 I/O에 대한 지연시간을 나타낸다.

도 9는 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존 환경에 대하여 네트워크 입출력 집약적인 가상머신의 네트워크 I/O 처리량을 나타낸다.

도 10은 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 입출력과 CPU 작업을 동시에 수행하는 가상머신의 네트워크 I/O 처리량을 나타낸다.

도 11은 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 입출력과 CPU 작업을 동시에 수행하는 가상머신의 디스크 I/O 대역폭을 나타낸다.

도 12는 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 입출력과 CPU 작업을 동시에 수행하는 가상머신의 디스크 I/O 대역폭을 나타낸다.

도 13은 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 입출력과 CPU 작업을 동시에 수행하는 가상머신에 대하여 CPU 작업에 대한 처리 시간을 나타낸다.

도 14는 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 타겟 가상머신을 정해 두어 입출력과 CPU 작업을 동시에 수행하는 다른 가상머신 하나씩 증가시켰을 때, 타겟 가상머신에 대한 CPU 작업 처리시간을 나타낸다.

도 15는 본 발명의 일 실시예에 따른 발명 기법이 적용된 환경과 적용되지 않은 기존환경에 대하여 타겟 가상머신을 정해 두어 입출력 작업을 수행하는 다른 가상머신을 하나씩 증가시켰을 때, 타겟 가상머신에 대한 CPU 작업 처리시간을 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0015] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0016] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0017] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0018] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0019] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0020] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0021] 이하, 첨부된 도면을 참조하여, 본 발명의 실시예들에 따른 크레딧 스케줄링 방법을 설명하기로 한다. 이하에서 설명되는 스케줄링 방법은 적어도 프로세서(processor) 및/또는 메모리(memory)를 포함하는 컴퓨팅 장치, 예컨대 서버에 의해 수행되는 동작들을 포함할 수 있다.

[0023] 도 1은 종래의 가상화 환경에서 I/O 요청 및 응답에 대한 수행 과정을 설명하기 위한 모식도이다.

[0024] 도 1을 참조하면, 가상화 환경에서 가상머신이 I/O를 발생시키면 공유 링 버퍼(I/O RING)에 I/O 요정에 대한 정보를 기입(도 1이 (a)) 후 드라이버 도메인(DOM0)으로 I/O 이벤트를 발생시키고(도 1의 (b)), 드라이버 도메인(DOM0)의 vCPU가 pCPU를 점유하면, 이벤트 채널(EVENT CHANNEL)을 확인 후 공유 링 버퍼에서 I/O 요청 정보에 대한 실질적인 I/O 핸들러를 수행한다(도 1의 (c)). 드라이버 도메인(DOM0)은 I/O 핸들러 수행을 완료하고, 공유 링 버퍼에 I/O 요청에 대한 I/O 응답을 기입한(도 1의 (d)) 후, I/O 요청을 한 가상머신의 이벤트 채널로 I/O 이벤트를 발생시킨다(도 1의 (e)). I/O 요청을 한 가상머신의 vCPU가 pCPU를 점유하면, 이벤트 채널을 확인 후, 드라이버 도메인으로부터 발생된 I/O 이벤트를 확인하면 공유 링 버퍼에서 I/O 응답에 대한 정보를 꺼내와 I/O 핸들러를 수행한다.

[0025] 도 2는 본 발명의 일 실시예에 따른 (크레딧) 스케줄링 방법을 설명하기 위한 순서도이다.

[0026] 도 2를 참조하면, 가상머신의 I/O 집약적 워크로드를 처리하는 vCPU에 대한 I/O 정보를 수집할 수 있다(S110).

[0027] 본 발명의 일 실시예에 따르면, I/O 또는 CPU 성능 측정 시, 외부 모니터링 모듈에 의존하지 않고, 스케줄링 매개 변수를 사용할 수 있다. 이는 외부 모니터링 모듈은 독립적으로 수행되므로 가상머신 스케줄러의 스케줄링 매개 변수와 동기화해야 하고, 드라이버 도메인은 불필요한 정보 조사와 가상머신의 지속적인 부하 추적으로 인해 추가적인 과부하를 추가하기 때문이다.

[0028] 또한, credit1 스케줄러에서 vCPU의 I/O 성능 정도는 스케줄링 영역의 부스트(BOOST) 메커니즘을 통해서 쉽게 알 수 있으나, credit2 스케줄러에는 부스트 메커니즘이 없기 때문에 스케줄러에서만 vCPU의 I/O 발생 정도를 알기가 어렵다. 전술한 외부 모니터링 모듈에서 발생하는 오버 헤드를 방지하고 credit2 스케줄러에서 I/O 발생을 알기 위해 본 발명의 일 실시예는 vCPU의 I/O 강도(strength)를 인식하는 이벤트 채널 메커니즘을 사용할 수 있다. credit2 스케줄러 정책에 따라 특정 런-큐(런-큐)에서 크레딧 재할당(credit reallocation)을 호출하면 공유 메모리 통신을 통해 특정 vCPU의 I/O 강도인 I/O 이벤트 정보를 가져와 스케줄러를 반영할 수 있다.

[0029] 도 3은 본 발명의 일 실시에에 따른 크레딧 스케줄러를 설명하기 위한 모식도이고, 도 4는 본 발명의 일 실시예에 따른 크레딧 스케줄러의 통합 vCPU 정보 구조체를 설명하기 위한 도면이다. 즉, 도 4는 본 발명의 일 실시예에 따른 크레딧 스케줄링 기법을 수행할 시, 이벤트 채널 영역(vCPU 이벤트 정보 구조체), 하이퍼바이저 스케줄러 영역(통합 vCPU 정보 구조체), 및 런-큐(런-큐 정보 구조체) 영역에 대한 구조체를 나타낸다.

[0030] 도 3을 참조하면, 드라이버 도메인(DOM0)은 다른 가상머신들로부터 성능 간섭을 배제하고, 드라이버 도메인만의 I/O 응답에 대한 I/O 이벤트 발생 정보를 얻기 위해 특정 물리 씨피유에 고정(pinning)시킬 수 있다.

[0031] 이벤트 체커(실시예에 따라 이벤트 체킹부, 이벤트 체킹 모듈, 이벤트 체커 모듈 등과 같은 상이한 명칭으로 명명될 수도 있음)는 어느 vCPU가 해당 이벤트를 처리할 것인지를 알아내는 것이 중요하며, 해당 vCPU가 어느 가상머신의 vCPU인지, 어느 런-큐에서 수행하고 있는지를 알기 위해 발명 기법의 통합 vCPU 구조체(도 4 참조)와 공유 메모리를 통항 양방향 통신을 수행할 수 있다. 구체적으로, 이벤트 체커는 가상머신들의 I/O 요청에 따른 드라이버 도메인이 특정 가상머신에게 I/O 응답을 알리기 위한 I/O 응답 이벤트 발생 횟수와 발생 주기 또는 간격(frequency or interval)을 도출하며, 도출된 I/O 이벤트 발생 정보를 가상화 환경에서 수행하고 있는 각 가상머신에 대하여 관리할 수 있다.

[0032] 도 3에서, 특정 런-큐에 모든 vCPU들에 대한 크레딧 재할당이 발생되면, 이벤트 체커로 특정 런-큐의 아이디(ID)를 보내어 해당 런-큐의 아이디를 가지고 있는 가상머신의 vCPU에 대한 I/O 이벤트 발생 정보를 초기화하고

다시 I/O 이벤트 발생 횟수와 주기를 측정할 수 있다.

[0033] 이벤트 체커는 본 발명 기법의 가상머신 스케줄러로부터 공유 메모리 통신을 통해 이벤트 체커가 관리하고 있는 vCPU 이벤트 정보 구조체에 런-큐 아이디(runq_id)를 매핑하며, 런-큐 아이디(runq_id)는 이벤트 체커가 크레딧 재할당이 어느 런-큐에서 발생했는지, 어느 vCPU의 vCPU 이벤트 정보를 초기화할 것인지를 추적하여 초기화한다.

[0034] 도 4를 참조하면, 이벤트 채널 영역(vCPU 이벤트 정보 구조체)과 (하이퍼바이저) 스케줄링 영역(통합 vCPU 정보 구조체, 스케줄러 영역이라 명명될 수도 있음)이 독립적으로 수행되므로 통합 vCPU 정보가 필요할 수 있다. 스케줄링 영역 및 이벤트 채널 영역에 있는 모든 스케줄링 정보는 크레딧 재할당이 발생하기 전에 최신 상태로 유지되는 통합 스케줄링 정보 구조일 수 있다. 또한, 크레딧 재할당은 런-큐에서 워크로드가 다른 vCPU의 배치 경향에 따라 발생하므로 스케줄링 영역에서 얻을 수 있는 런-큐 정보(런-큐 정보 구조체)가 추가될 수 있다.

[0035] 한편, 도 3을 참조하면, 드라이버 도메인은 다른 가상머신들로부터의 성능 간섭을 배제하고, 드라이버 도메인만의 I/O 응답에 대한 I/O 이벤트 발생 정보를 얻기 위해 이벤트 채널에서 발생되는 드라이버 도메인과 가상머신 간의 I/O 응답 이벤트에 대한 정보를 추적 및 분석할 수 있다.

[0036] 도 2를 참조하면, 상기 수집된 I/O 정보를 통해 I/O 워크로드 정도에 따라 매핑된 I/O 가상 크레딧(I/O virtual credit: IVCredit)을 I/O 집약적 vCPU에 할당할 수 있다(S120).

[0037] 도 3의 하이퍼바이저 스케줄러는 특정 코어의 런-큐에서 크레딧 재할당이 발생하면 이벤트 체커로부터 vCPU 이벤트 정보를 공유 메모리 통신을 통해 받아와 통합 vCPU 정보 구조체의 정보를 최신화할 수 있다. 하이퍼바이저 스케줄러는 통합 vCPU 정보 구조체의 정보를 이용하여 각 vCPU의 IVCredit과 CVCredit을 할당할 수 있다.

[0038] 우선, 본 발명이 일 실시예에서, vCPU의 I/O 리소스 점유 보장은 I/O 강도를 고려하여 I/O 집약적인 vCPU에 할당되는 IVCredit의 양일 수 있다. I/O 집약적 vCPU에서 I/O 강도를 IVCredit에 매핑하는 첫 번째 작업은 드라이브 도메인에서 발생한 모든 I/O 이벤트에서 평균 I/O 이벤트 간격을 가져오는 것일 수 있다. 이벤트 채널에서 드라이버 도메인은 특정 vCPU로의 I/O 응답 이벤트 비트를 설정하여 비동기적으로 I/O 응답 이벤트를 발생시켜 상기 특정 vCPU가 I/O 응답 이벤트를 처리하는 일반적인 가상화 시스템의 비동기적 I/O 처리 과정에서 드라이버 도메인으로부터 발생 되는 I/O 응답 이벤트를 추적 및 분석하여 가상머신의 특정 vCPU로 I/O 요청 완료를 알리는 I/O 이벤트의 평균 간격을 획득할 수 있다. 드라이버 도메인은 모든 게스트 가상머신의 I/O 요청을 처리할 수 있다. 따라서 드라이브 도메인에서 게스트 가상머신에 대한 I/O 응답 완료를 알리기 위해 발생하는 I/O 이벤트 간의 시간 간격은 게스트 가상머신 중 하나의 I/O 요청에 대한 드라이브 도메인 처리 시간이라고 할 수 있다. 하기의 수학식 1을 기반으로, 드라이브 도메인에서 발생한 모든 I/O 이벤트에 대한 평균 시간 간격인 $\overline{I_e}$ 을 계산할 수 있다.

[0039] [수학식 1]

[0040] $$\overline{I_e} = 0.9 \times I_e + (1 - 0.9) \times \overline{I_{e-1}}, \quad where \;\; t_e - t_{e-1} \;\; is \;\; I_e$$

[0041] 수학식 1에서, 지수 이동 평균(EMA) 기반으로 각 vCPU들에 대하여 평균 I/O 응답 이벤트 발생 정도를 측정한 것이며, EMA에 국한하지 않고, 가중치 이동 평균, 단순 이동 평균 등 다양한 평균 공식을 적용할 수 있다.

[0042] 수학식 1에서, t_e 는 현재 드라이버 도메인으로부터 I/O 응답 이벤트를 받은 시각을 나타낸다. $t_e - t_{e-1}$는 드라이버 도메인으로부터 전송된 I/O 응답 이벤트에 대한 주기를 의미하여 I_e 로 표현될 수 있다.

[0043] 수학식 1을 통해서 EMA가 적용되는 한 개의 vCPU에 대하여 I/O 응답 이벤트의 평균 주기에 대해 최근 I/O 발생 동향이 적용된 I/O 응답 이벤트 평균 주기를 얻을 수 있다.

[0044] 수학식 1에 따르면, 드라이브 도메인에서 발생한 모든 I/O 이벤트에 대해 최신의 I/O 응답 발생 주기를 도출하여 vCPU의 입출력 강도를 정의할 수 있는 메트릭으로 이용될 수 있다.

[0045] 수학식 1에 대하여, $\overline{I_e}$ 은 드라이브 도메인이 특정 vCPU로 발생되는 I/O 응답 이벤트 평균 주기에 대하여 런-큐에서 크레딧 재할당이 호출 될 때마다 다시 초기화될 수 있다.

[0046] $\overline{I_e}$ 을 기반으로 가상머신의 각 vCPU에 대한 IVCredit을 계산할 수 있다. 하기의 수학식 2를 참조한다.

[0047] [수학식 2]

$$IVCredit = E_n \times \overline{I_e} \times \frac{consumed\ credit}{\frac{\sum_i^n cc_i}{n}}$$

[0048]

[0049] 수학식 2에서, E_n 은 런-큐에서 크레딧 재할당이 발생하기 전까지의 특정 vCPU가 드라이버 도메인으로부터 수신한 I/O 응답 이벤트의 개수를 뜻하며, $\overline{I_e}$ 는 상기 수학식 1에서 EMA가 적용된 I/O 응답 이벤트에 대한 평균 시간 간격을 의미할 수 있다.

[0050] 수학식 2에서, n 은 런-큐에 배치된 vCPU의 총 개수를 뜻하며, cc_i 는 런-큐 내부에 있는 vCPU에 대하여 크레딧 재할당이 수행되기 전에 소비된 크레딧의 양을 의미하고, $consumed\ credit$ 는 IVCredit을 할당할 vCPU가 실제 런-큐에서 크레딧 재할당이 호출 될 때 까지 소비된 크레딧의 양을 의미한다.

[0051] 수학식 2에서, $E_n \times \overline{I_e}$ 는 vCPU에 대해 드라이버 도메인으로부터 수신한 I/O 응답 이벤트 수와 EMA가 적용된 I/O 이벤트의 평균 간격을 곱하는 것을 표현한 것이며, 이는 런-큐 내 vCPU 중에서 특정 vCPU의 차별화된 I/O 강도를 도출한다.

[0052] 수학식 2에서, CPU 집약적 vCPU의 값은 드라이버 도메인(DOM0)으로부터 I/O 응답 이벤트를 수신하거나 드라이버 도메인(DOM0)에 I/O 요청 이벤트를 생성할 가능성이 낮기 때문에 I/O 집약적 vCPU보다 상대적으로 IVCredit 값이 0 또는 상대적으로 적은 값을 가지므로, CPU 집약적인 워크로드를 가진 vCPU와 I/O 집약적인 워크로드를 가진 vCPU를 구별할 수 있다.

[0053] 수학식 2에서, $\frac{consumed\ credit}{\frac{\sum_i^n cc_i}{n}}$ 는 각 vCPU들에 대하여 실질적인 작업 정도에 따른 vCPU 만의 크레딧 소비량을 IVCredit 양에 반영하기 위해 곱하는 과정을 포함할 수 있으며, 이를 통해 실질적인 런-큐 내부의 vCPU들에 대해서 크레딧 소비량에 따른 차등화된 IVCredit을 할당 할 수 있는 단계를 의미할 수 있다.

[0054] 수학식 2에서, IVCreidt은 가상머신의 vCPU들 중, I/O을 수행하기 위한 vCPU의 pCPU점유를 보장시킬 수 있다. 또한, 코어의 런-큐에 할당된 각 가상머신의 vCPU에 대하여 I/O 발생 정도에 따라 차등화된 IVCredit을 할당할 수 있다.

[0055] 수학식 2를 참고하면, CPU-집약적인 작업을 수행하는 가상머신의 vCPU는 I/O 요청이 비교적 I/O-집약적인 작업을 수행하는 가상머신의 vCPU보다 적어 IVCreidt을 더 적게 받음으로, I/O를 수행하는 vCPU는 CPU작업을 수행하는 vCPU보다 pCPU 점유를 더 보장받을 수 있다.

[0056] I/O-집약적인 vCPU가 IVCredit으로 인해 pCPU 점유를 보장받음으로써 나타나는 CPU-집약적인 vCPU의 CPU 처리 성능 저하 방지를 위하여 CPU 집약적 vCPU에 CPU 가상 크레딧(CPU virtual credit: CVCredit)을 할당할 수 있다(S130).

[0057] 수학식 1과 수학식 2를 참고하면, 특정 런-큐에서 크레딧 재할당이 발생되면 각 실행 대기열 내의 I/O 집약적인 vCPU에게 IVCredit이 제공될 수 있다. 또한, 크레딧 재할당 후, I/O 집약적 vCPU는 CPU 집약적 vCPU보다 더 많은 크레딧을 갖게 되므로 I/O-집약적 vCPU는 CPU-집약적 vCPU보다 pCPU를 점유할 수 있는 기회가 높게 된다. 그러나 I/O 집약적 vCPU에 IVCredit을 할당하면 CPU-집약적 vCPU에서 CPU 성능 저하가 발생할 수 있으므로, CPU-집약적인 vCPU의 CPU 성능 저하를 방지해야 한다.

[0058] [수학식 3]

$$\frac{\sum_{i=1}^n IVCredit_i}{k} \times \frac{1}{IVCredit_c + 1} > 1$$

[0059]

[0060] 수학식 3은 CVCredit을 할당하기 위한 vCPU를 정의하는 기준을 나타낸다.

[0061] 수학식 3에서, IVCredit은 수학식 2에 의해 할당된 vCPU의 IVCredit 양을 의미하고, k는 런-큐에 있는 vCPU의 수를 의미한다.

[0062] 수학식 3을 참고하면, 런-큐에서 IVCredit이 상대적으로 높은 vCPU는 1 미만의 값을 가지게 되며, IVCredit이 상대적으로 낮은 vCPU는 1을 초과하는 값을 가지게 된다.

[0063] 수학식 3을 충족하면(좌측 값이 1보다 크면) 해당 vCPU는 CPU-부스트 그룹에 추가한다. 수학식 3을 충족하지 않으면(좌측 값이 1보다 작으면) 해당 vCPU는 CPU-부스트 그룹에 추가하지 않는다. CPU-부스트 그룹은 수학식 3을 충족하는 CPU 집약적 vCPU들의 풀이며 런-큐 단위로 관리된다.

[0064] 수학식 3에서, CVCredit은 런-큐의 재할당을 호출한 vCPU에게 할당되며, 런-큐의 크레딧 재할당을 호출한 vCPU는 할당된 크레딧이 0이 되었을 때 CVCredit을 할당하여 지속적인 vCPU의 CPU작업을 수행하도록 한다.

[0065] [수학식 4]

[0066] $$CVCredit = \frac{\sum_{i=1}^{n} IVCredit_i}{j} \times \frac{consumed\ credit}{default\ credit}$$

[0067] 수학식 4은 특정 런-큐에서 CPU-부스트 그룹에 포함된 vCPU들에 CVCredit을 할당하는 과정을 표현한다. 수학식 4에서, j는 CPU-BOOST 그룹에 속하는 vCPU의 수를 의미하고 n은 런-큐에 있는 모든 vCPU의 수를 의미한다.

[0068] 수학식 4에서, consumed credit은 런-큐에서 크레딧 재할당이 발생하기까지의 실질적인 vCPU가 pCPU를 점유하여 소비한 크레딧의 양이고, default credit은 크레딧 재할당이 발생할 때, 모든 vCPU에 대하여 동등한 pCPU 점유 시간을 보장하기 위한 정적 크레딧 양을 표현한다.

[0069] $\frac{\sum_{i}^{n} IVCredt_i}{j}$ 는 런-큐 내에서 I/O-집약적인(IVCredit이 높은) vCPU에 할당된 평균 IVCredit의 양을 나타낸다. 수학식 4에서, $\frac{consumedcredit}{defaultcredit}$ 는 vCPU에 대한 실제 CPU 작업 정도에 따른 차별화된 CVCredit을 위해 곱해진다.

[0070] 수학식 3과 수학식 4를 통해 구해준 CVCredit은 스케줄링될 다음 vCPU가 크레딧 재할당 조건을 충족하면 CPU-부스트 그룹에 포함된 vCPU만 스케줄링되는 CPU-부스트 상태로 전환된다. CVCredit은 CPU-BOOST 상태에서, 다음 스케줄링될 CPU-부스트 그룹에 포함된 vCPU의 CVCredit이 0 미만이 되면 런-큐의 모든 vCPU에 대해 크레딧 재할당이 발생한다.

[0071] 수학식 1 내지 수학식 4에 대해서, I/O 집약적 vCPU에 IVCredit을 할당하기 때문에 CPU 집약적 vCPU에서 성능 간섭이 발생할 수 있어서, CPU 집약적 vCPU의 성능 간섭을 피하기 위해 크레딧 재할당을 호출한 vCPU에 CVCredit을 할당할 수 있다 또한, CVCredit은 런-큐에 있는 모든 vCPU에 대한 IVCredit을 기반으로 할당되며 크레딧 재할당을 호출한 vCPU의 크레딧이 0 일 때 CPU-부스트 상태일 때 할당된 CVCredit을 소비할 수 있다.

[0072] 본 발명의 일 실시예를 활용하면, I/O 집약적인 vCPU의 pCPU를 선점할 기회를 높일 수 있다. 크레딧을 할당할 때 런-큐에서 크레딧 재할당을 호출 될 때까지 수집 된 vCPU의 정보를 통해 IVCredit 및 CVCredit을 계산할 수 있다. 크레딧 재할당을 마치면, I/O 집약적인 vCPU는 CPU 집약적인 vCPU보다 더 많은 크레딧을 가질 수 있다. 따라서 IO 집약적 vCPU는 CPU 집약적 vCPU보다 pCPU 점유 기회가 더 높을 수 있다.

[0073] 도 5는 본 발명의 일 실시예에 따른 크레딧 스케줄링 결과를 설명하기 위한 도면이다. 구체적으로, 도 5는 본 발명의 일 실시예에 따른 스케줄링 방법에서 하이퍼바이저 스케줄러에 적용되는 I/O 가상크레딧과 CVCredit을 통한 가상크레딧 할당 기법과 크레딧 은닉 기법으로써, 각 런-큐의 vCPU들에 대하여 크레딧 재할당이 발생되고 난 후의 가상 크레딧 할당 기법과 크레딧 은닉 기법을 도시하고 있다.

[0074] 도 5에서, 본 발명 기법은 특정 코어의 런-큐에서 크레딧 재할당이 발생하면 이벤트 체커로부터 vCPU 이벤트 정보를 공유 메모리 통신을 통해 받아와 통합 vCPU 정보 구조체의 정보를 최신화한다.

[0075] 도 5에서, 본 발명 기법의 하이퍼바이저 스케줄러는 통합 vCPU 정보 구조체의 정보를 이용하여 각 vCPU의 IVCredit과 CVCredit을 할당한다.

[0076] 도 6은 도 5의 크레딧 은닉 기법을 통한 가상 크레딧(IVCredit, CVCreidt)을 할당받은 vCPU들에 대한 런-큐 배

치 상태를 도시화한것이다.

[0077] 도 5는 수학식 1 내지 수학식 4를 통해 도출해낸 IVCredit과 CVCredit이 할당된 vCPU의 크레딧의 양을 의미할 수 있다. 도 5에서, vCPU1, vCPU2, vCPU3은 vCPU4와 vCPU5 보다 IVCredit 할당으로 인해 더 많은 양의 크레딧을 가지고 있지만, 크레딧 재할당이 *vCPU.endcredit*으로 인해 기존 크레딧 재할당 호출 조건(다음 스케줄링될 vCPU의 크레딧 양이 0일 때)이 *0 + IVCredit*으로 바뀌어 IVCredit을 받지 않는 vCPU들과 최대한의 같은 양의 크레딧을 재할당할 수 있다.

[0078] 도 6은 본 발명 기법에서 크레딧 은닉 기법을 통해 정렬된 코어의 런-큐 상태를 도시화한 것이다.

[0079] 도 6에서 상위 런-큐 상태는 크레딧 재할당 이후 IVCreidt을 할당받아 크레딧의 양이 많은 대로 배치된 상태(vCPU1, vCPU2, vCPU3, vCPU4, vCPU5 순)를 의미하고, 하위 런-큐 상태는 CPU-부스트 상태로 전환되었을 때의 코어의 런-큐 배치상태를 나타낸다.

[0080] 도 6을 참고하면, I/O-집약적인 vCPU의 I/O 강도가 높은 vCPU에게 IVCredit을 더 많이 할당하여 pCPU를 먼저 선점할 수 있는 상황, CPU-부스트 상태에서는 CPU 작업 정도에 따라 CVCredit을 많이 할당받은 vCPU 순으로 vCPU를 스케줄링하는 상황, 및 vCPU3는 IVCredit을 할당받았지만, 수학식 3을 만족시킴으로 CVCredit을 할당받은 상황을 도시화한 것이라 할 수 있다.

[0081] 하기의 알고리즘 1은 본 발명의 일 실시예에 따른 가상머신 스케줄링 방법에 사용될 수 있다.

[0082] [알고리즘 1]

Algorithm 1 ISACS: Scheduling algorithm

$vCPU_c$: Current vCPU on the pCPU

$vCPU_n$: Next vCPU to Run

1: **if** $CPU - BOOST\ STATE == FALSE$ **then**
2: $Burn\ the\ vCPU_c.credit$
3: $Set\ vCPU\ with\ highest\ credits\ for\ all\ vCPUs\ in\ the\ RunQ\ as\ vCPU_n$
4: **if** $vCPU_n.credit \leqq vCPU_n.endcredit$ **then**
5: $CPU - BOOST\ STATE \leftarrow TRUE$
6: $Set\ vCPU\ with\ highest\ CVCredits\ over\ all\ vCPUs\ in\ CPU - BOOST\ GROUP\ as\ vCPU_c$
7: **end if**
8: **end if**
9: **if** $CPU - BOOST\ STATE == TRUE$ **then**
10: $Burn\ the\ vCPU_c.CVCredit$
11: $Set\ vCPU\ with\ highest\ CVCredits\ over\ all\ vCPUs\ in\ CPU - BOOST\ GROUP\ as\ vCPU_n$
12: **if** $vCPU_n.credit \leqq vCPU_i.credit$ **then**
13: $RunQ.IOload \leftarrow 0$
14: $Init\ the\ member\ of\ CPU - BOOST\ GROUP$
15: $CPU - BOOST\ STATE \leftarrow FALSE$
16: **for** $vCPU_i \in RunQ$ **do**
17: $Derive\ IVCredit\ for\ vCPU_i\ using\ Equation\ 1\ and\ Equation\ 2$
18: $vCPU_i.credit \leftarrow default\ credit + IVCredit$
19: $vCPU_i.endcredit \leftarrow IVCredit$
20: $RunQ.IOload+ = IVCredit$
21: **end for**
22: **for** $vCPU_i \in RunQ$ **do**
23: **if** $Equation\ 3\ is\ satisfied$ **then**
24: $Join\ vCPU_i\ to\ the\ CPU - BOOST\ GROUP$
25: $vCPU_i.CVCredit \leftarrow CVCredit\ which\ is\ calculated\ by\ using\ Equation\ 4$
26: **end if**
27: **end for**
28: **end if**
29: **end if**
30: *return* $vCPU_n$

[0083]

[0084] 알고리즘 1에서, 전체적인 I/O-집약적인 vCPU들에 대하여 수학식 1와 수학식 2를 통해 IVCredit을 할당하고, 전체적인 CPU-집약적인 vCPU들(IVCredit 할당을 상대적으로 적게 받은 vCPU)에 대하여 수학식 3을 통하여 CPU-BOOST 그룹에 포함시키며, 수학식 4를 통해 CVCredit을 할당하는 것을 알 수 있다.

[0085] 알고리즘 1을 참조하면, 다음 스케줄링될 vCPU의 크레딧이 자신의 end.credit 보다 적을 때 CPU-BOOST 상태가 되어 CPU-BOOST 그룹 내에 있는 vCPU들의 CVCredit을 소비하는 것을 알 수 있다. 또한, 특정 런-큐에서 다음 스케줄링될 vCPU의 CVCredit이 0과 같거나 작으면 크레딧 재할당이 발생하여 IVCredit과 CVCredit이 할당되는 것을 확인할 수 있다.

[0086] 본 발명의 일 실시예에 따르면, 런-큐의 모든 vCPU에 대한 크레딧 재할당이 완료된 후, 하기의 수학식 5, 수학

식 6, 및 알고리즘 2를 기반으로, 각 런-큐의 I/O 워크로드 정도에 따라 vCPU의 마이그레이션을 수행하는 단계를 더 포함할 수 있다.

[0087] pCPU간에 균등한 I/O 워크로드 분산을 제공하기 위해 vCPU를 마이그레이션할 때 마이그레이션되는 vCPU의 대상이 있는 런-큐의 상태에 주의해야 한다. 런-큐의 모든 vCPU에서 수신한 총 I/O 이벤트 수와 런-큐의 I/O로드 정도를 마이그레이션 조건으로 취할 수 있다.

[0088] [수학식 5]

[0089] $$RunQ_i.IOevent < RunQ_s.IOevent$$

[0090] 수학식 5는 하기의 알고리즘 2에서 vCPU 마이그레이션을 수행할 타겟 코어의 런-큐를 도출하는 과정을 수식화한 것이다. 수학식 5에서, $RunQ_i$ 는 크레딧 재할당이 완료된 특정 코어의 런-큐를 제외한 vCPU 마이그레이션을 수행할 타겟 코어의 런-큐를 의미하고, $RunQ_s$ 는 크레딧 재할당이 완료된 특정 코어의 런-큐를 의미한다. 즉, 수학식 5는 크레딧 재할당이 완료된 특정 코어의 런-큐를 제외한 I/O 로드 밸런싱을 수행하기 위한 타겟 코어의 런-큐를 도출하는 과정을 나타낸다. $.IOevent$ 는 각 런-큐에 배치된 모든 vCPU에 대하여 도메인 드라이버로부터 수신한 I/O 응답 이벤트 개수에 대한 총합을 표현한다.

[0091] 수학식 5을 만족하면, $RunQ_i$ 는 vCPU 마이그레이션을 수행할 타겟 코어의 런-큐로 지정한다.

[0092] [수학식 6]

[0093] $$RunQ_d.IOload + vCPU_j.IVCredit < RunQ_s.IOload - vCPU_j.IVCredit$$

[0094] 수학식 6은 도출된 타겟 코어의 런-큐로 보낼 vCPU를 지정하는 과정을 수식화 한 것이다. 수학식 6에서, $vCPU_j$ 는 RunQd로 마이그레이션을 수행할 런-큐에 배치된 모든 vCPU들 중 하나일 수 있으며, $RunQd.IOload$는 vCPU가 마이그레이션될 타겟 코어의 런-큐를 의미하고, $RunQs.IOload$는 마이그레이션이될 vCPU를 가지고 있고, 크레딧 재할당을 완료한 코어의 런-큐를 의미할 수 있다.

[0095] 수학식 6에서, .IOload는 런-큐 내부의 모든 vCPU들에게 할당된 vCPU의 IVCredit에 대한 총합을 나타내며, IVCredit은 본 발명 실시예에 따른 vCPU에 대한 I/O 강도를 나타냄으로 IOload가 높을수록 I/O 강도가 높은 vCPU들이 많이 배치된 런-큐라고 인식할 수 있다.

[0096] 수학식 6에서, vCPU 마이그레이션을 수행 시, vCPU가 할당된 코어의 런-큐의 I/O 로드와 타겟 코어의 런-큐 I/Oload를 비교하여 타겟 코어의 런-큐의 I/Oload가 vCPU가 할당된 코어의 런-큐의 I/Oload 를 넘지 못하도록 하는 것이 중요하다.

[0097] [알고리즘 2]

[0098]

Algorithm 2 ISACS: I/O load-balancing

```
for RunQ_i ∈ RunQ do
  if RunQ_i.id == RunQ_s.id then
    continue
  end if
  if RunQ_i.IOevent < RunQ_s.IOevent then
    RunQ_d ← RunQ_i
  end if
end for
for vCPU_j ∈ RunQ_s do
  if RunQ_d.IOload + vCPU_j.IVCredit < RunQ_s.IOload − vCPU_j.IVCredit then
    vCPU_m ← vCPU_j
  end if
end for
if vCPU_m!=NULL then
  Migrate vCPU_m to RunQ_d
  Return
end if
Return
```

[0100] 이하, 본 발명의 일 실시예에 따른 크레딧 스케줄링 방법이 적용된 가상화 환경과 종래의 가상화 환경에 대하여 가상머신 구성에 따른 네트워크 I/O 성능을 확인 및 비교하는 실험 결과를 설명하기로 한다.

[0101] 도 7 내지 도 15는 본 발명 실시예에 따른 본 발명 기법이 적용된 환경과 기존 환경에 대한 I/O 성능 평가 및 CPU 성능 평가를 실제 진행한 결과 값을 도시화 한 것이다.

[0102] 도 7 내지 도 10과 도 13 내지 도 15에서, 드라이버 도메인은 pCPU0과 pCPU1에 고정(pinning) 할당하였고, 클라이언트 가상머신은 pCPU 7에 고정 할당한다. 그 외 나머지 게스트 가상머신(드라이번 도메인, 클라이언트 가상머신이 아닌 가상머신)은 pCPU2 내지 pCPU6에 고정 할당하였다. 또한, 클라이언트 가상머신은 드라이버 도메인을 제외한 나머지 게스트 가상머신에게 TCP/IP 통신을 수행하며, 게스트 가상머신들은 TCP/IP 통신에 대한 네트워크 I/O을 수신한다.

[0103] 도 11 내지 도 12는 드라이버 도메인은 pCPU0와 pCPU1에 고정 할당하고, 나머지 게스트 가상머신들은 pCPU2내지 pCPU7에 고정할당하였다.

[0104] 도 7 내지 도 15에 대한 각 실험에 대하여, 가상머신 구성에 따라 각 라운드당 15번의 실험을 진행한 결과에 대한 평균 값을 도출한 실험 결과 값이다. 도 7 내지 도 15에서, i는 I/O집약적인 작업을 처리하는 가상머신, c는 CPU집약적인 작업을 처리하는 가상머신, m은 CPU 집약적인 작업과 I/O 집약적인 작업을 동시에 처리하는 가상머신을 의미한다.

[0105] 도 7과 도 8에 대하여 pCPU7에 할당된 클라이언트 가상머신이 게스트 가상머신들에게 TCP/IP 통신을 수행하는 동시에 게스트 가상머신들에 대한 네트워크 지연시간을 측정하였다.

[0106] 도 7 내지 도 8은 pCPU7에 할당된 클라이언트 가상머신에 대하여 pCPU2부터 pCPU6에 할당된 가상머신에 대한 네트워크 지연시간을 실제 측정한 실험 결과 값이다.

[0107] 도 9 내지 도 12에서, 'm3'은 CPU 집약적인 작업과 I/O 집약적인 작업을 동시에 처리하는 가상머신 3개다 가상화 환경에서 실험을 실제 수행한 실험 결과값, 'i1c11'은 I/O 작업을 처리하는 가상머신 1개와 CPU 작업을 처리하는 가상머신 11개가 수행되는 가상화 환경에서 실험을 실제 수행한 실험 결과값을, 'i5c7'은 I/O 작업을 처리하는 가상머신 5개와 CPU 작업을 처리하는 가상머신 7개가 수행되는 가상화 환경에서 실험을 실제 수행한 실험 결과값을 의미한다.

[0108] 도 7은 총 12개의 가상머신에 대하여 네트워크 I/O 집약적인 작업을 수행하는 가상머신에 대하여 지연 시간에 대한 결과 값을 나타낸다. 도 7에 대하여, (a)은 네트워크 I/O 집약적인 작업을 수행하는 가상머신 1개와 CPU 집약적인 작업을 수행하는 가상머신 11대를, (b)는 네트워크 I/O 집약적인 작업을 수행하는 가상머신 2개와 CPU 집약적인 작업을 수행하는 가상머신 10대를, (c)는 네트워크 I/O 집약적인 작업을 수행하는 가상머신 3개와 CPU 집약적인 작업을 수행하는 가상머신 9개를 의미한다.

[0109] 도 7은 총 1000번의 실험 결과값에 대하여 25%에서부터 75%까지의 범위에 대한 500개에 대한 네트워크 지연시간 분포 결과값을 나타낸다. 도 7을 참조하면, 기존 환경은 본 발명 기법보다 긴 꼬리 대기시간(tail-latency)을 가지고 있으며, 본 발명기법이 적용된 가상화 환경은 기존 환경보다 더 적은 레이턴시를 가지고 있음을 확인할 수 있다.

[0110] 도 8은 CPU와 I/O 집약적인 작업을 동시에 처리하는 가상머신 6대에 대한 2000번의 네트워크 지연시간 측정 결과 값에 대하여 25%에서부터 75%까지의 범위에 대한 1000번의 실험에 대한 대한 결과 값 분포를 도시화한 것이다. 도 8을 참조하면, 본 발명 기법이 적용된 가상화 환경은 기존환경보다 다 적은 네트워크 지연시간을 가지고 있음을 확인할 수 있다.

[0111] 도 9는 본 발명의 일 실시예에 따라 네트워크 I/O-집약적인 작업 또는 CPU 집약적인 작업을 수행하는 게스트 가상머신들에 대하여 발명 기법이 적용된 환경과 기존 가상화 환경에 대하여 가상머신 구성에 따른 네트워크 I/O 성능을 확인 및 비교하기 위한 실험 결과를 도시화한 것이다. 도 9를 참조하면, 11대의 CPU-집약적인 가상머신과 9대의 네트워크 I/O-집약적인 가상머신 3대가 수행되는 환경에서 기존 환경보다 발명 기법이 적용된 가상화 환경의 네트워크 I/O 처리량이 더욱 높은 것을 확인할 수 있다. 또한, 전체적인 실험 결과에 대하여 기존 가상화 환경보다 발명 기법이 적용된 가상화 환경의 네트워크 I/O 처리량이 높고, 편차가 적으므로 이에 의거하여 발명 기법이 적용된 가상화 환경의 네트워크 I/O 성능이 더 높은 것을 확인할 수 있다.

[0112] 도 10은 네트워크 I/O 와 CPU 작업을 동시에 수행하는 게스트 가상머신들에 대하여 기존 가상화 환경과 본 발명 기법이 적용된 가상화 환경에서의 네트워크 I/O 성능을 확인 및 비교하기 위한 실험 결과를 도시화한 것이다. 도 10을 참조하면, 기존 가상화 환경의 네트워크 I/O 처리량이 본 발명기법이 적용된 환경보다 전체적으로 낮으며, 발명 기법이 적용된 가상화 환경에서의 네트워크 I/O 처리량이 기존 가상화 환경보다 더 높은 네트워크 I/O 처리량을 가지고 있으며, 본 발명 기법이 적용된 가상화 환경에서의 네트워크 I/O 처리량이 더 높은 것을 확인할 수 있다.

[0113] 도 11은 본 발명의 일 실시예에 따라 디스크 I/O-집약적인 작업 또는 CPU 집약적인 작업을 수행하는 게스트 가상머신들에 대하여 발명 기법이 적용된 환경과 기존 가상화 환경에 대하여 가상머신 구성에 따른 네트워크 I/O 성능을 확인 및 비교하기 위한 실험 결과를 도시화한 것이다. 도 11을 참조하면, 전체적인 디스크 I/O 대역폭이 디스크 입출력 대상 파일과도 상관없이 기존 환경보다 본 발명 기법이 적용된 환경이 더 높은 것을 확인할 수 있다.

[0114] 도 12는 네트워크 I/O 와 CPU 작업을 동시에 수행하는 게스트 가상머신들에 대하여 기존 가상화 환경과 본 발명 기법이 적용된 가상화 환경에서의 디스크I/O 성능을 확인 및 비교하기 위한 실험 결과를 도시화한 것이다. 도 12를 참조하면, 전체적인 디스크 I/O 대역폭이 디스크 입출력 대상 파일과도 상관없이 기존 환경보다 본 발명 기법이 적용된 환경이 더 높은 것을 확인할 수 있다.

[0115] 도 13은 도 11과 같이 네트워크 I/O-집약적인 작업과 CPU-집약적인 작업을 동시에 수행하는 게스트 가상머신들이 수행되는 가상화 환경에서, 발명 기법이 적용됨으로 인한 CPU 성능 저하 정도를 인지하기 위하여 기존 환경과 비교를 통한 실험 결과를 도시화한 것이다. 도 13을 참조하면, 가상화 환경에서 1대의 가상머신부터 시작하여 총 6대가 수행할 때까지 1대씩 가상머신을 늘리며 실제 실험을 수행함을 알 수 있다. 도 13을 참조하면, 발명 기법이 적용된 가상화 환경에서 CPU 작업 처리시간에 대하여 기존 환경보다 본 발명 기법이 적용된 환경에 대한 CPU 작업 처리 시간이 조금 높은 것을 확인할 수 있지만, 이는 미세한 차이이므로 발명 기법 적용으로 인한 CPU 성능 하락 정도는 무시할 수 있다.

[0116] 도 14와 도 15는 1대의 타겟 가상머신을 수행시키면서 타겟 가상머신이 아닌 가상머신을 1대부터 타겟 가성머신을 제외한 총 5대의 가상머신이 수행될 때까지 가상머신을 1대씩 증가시킨 것을 확인할 수 있으며, 타겟 가상머신의 CPU 작업 처리 시간을 측정한 실험의 결과를 도시화 한 것이다. 도 14와 도 15에 대하여, 모든 실제 실험 결과는 타겟 가상머신 1대의 CPU 작업 처리 시간을 측정한 실험의 결과를 도시화한 것이다.

[0117] 도 14는 네트워크 I/O 작업만을 처리하는 가상머신을 증가시켰을 때의 실험 결과를 도시화한 것이다. 도 14를 참조하면, 기존 환경과 본 발명 기법이 적용된 환경에 대하여 미세한 차이를 확인할 수 있다.

[0118] 도 15는 I/O와 CPU 작업을 동시에 처리하는 가상머신을 증가시켰을 때의 실험 결과를 도시화한 것이다. 도 15를 참조하면, 발명 기법이 적용된 가상화 환경에서 CPU 작업 처리시간에 대하여 기존 환경보다 본 발명 기법이 적용된 환경에 대한 CPU 작업 처리 시간이 조금 높은 것을 확인할 수 있지만, 이는 미세한 차이이므로 발명 기법 적용으로 인한 CPU 성능 하락 정도는 무시할 수 있다.

[0119] 도 13 내지 도 15를 참조하면, 본 발명기법에서 IVCredit을 할당함으로써 발생할 수 있는 I/O-집약적인 vCPU의 pCPU 점유 보장으로 인해 나타날 수 있는 CPU-집약적인 vCPU의 CPU 성능 하락이 미비하다는걸 확인할 수 있다.

[0121] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

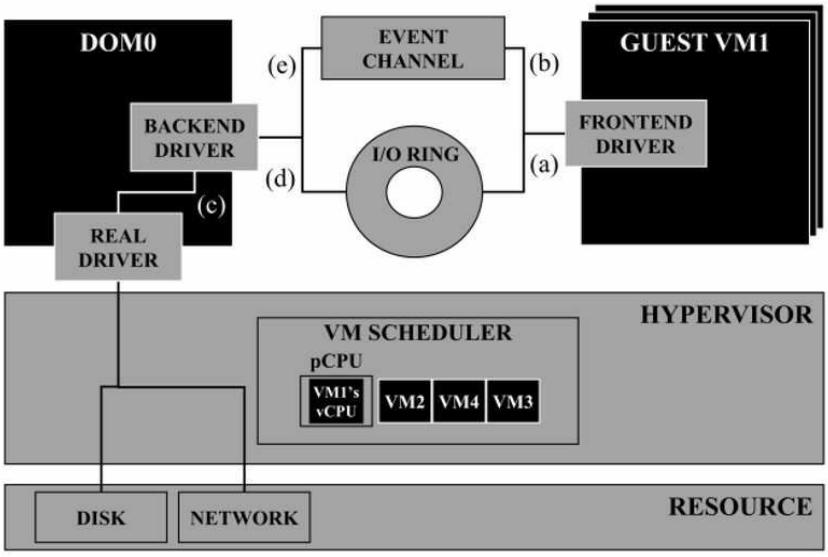
[0122] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0123] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

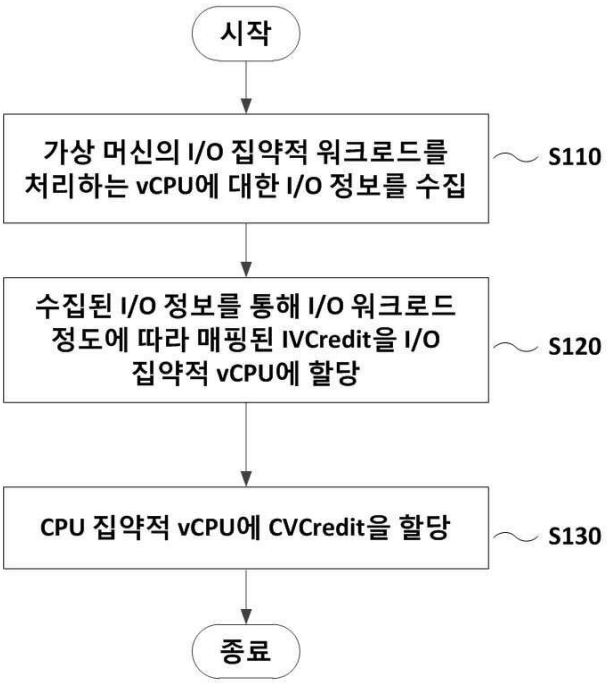
[0124] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성 요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

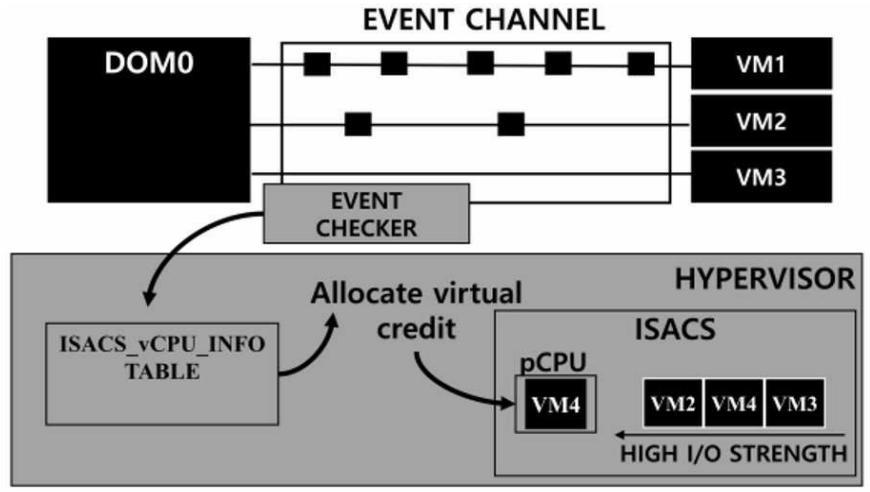
도면1



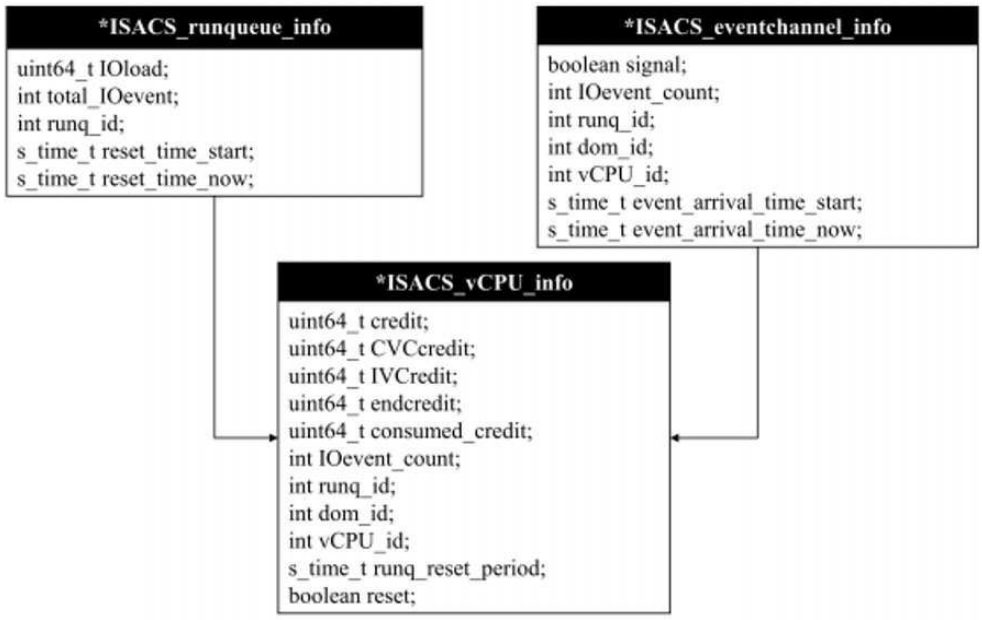
도면2


시작
가상 머신의 I/O 집약적 워크로드를 처리하는 vCPU에 대한 I/O 정보를 수집
S110
수집된 I/O 정보를 통해 I/O 워크로드 정도에 따라 매핑된 IVCredit을 I/O 집약적 vCPU에 할당
S120
CPU 집약적 vCPU에 CVCredit을 할당
S130
종료

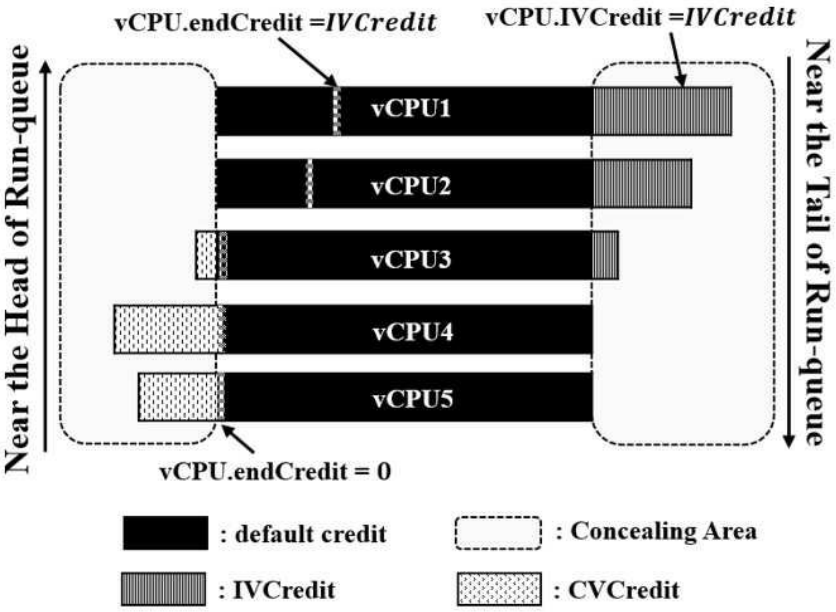
도면3


EVENT CHANNEL
DOM0
VM1
VM2
VM3
EVENT CHECKER
HYPERVISOR
Allocate virtual credit
ISACS_vCPU_INFO TABLE
ISACS
pCPU
VM4
VM2
VM4
VM3
HIGH I/O STRENGTH

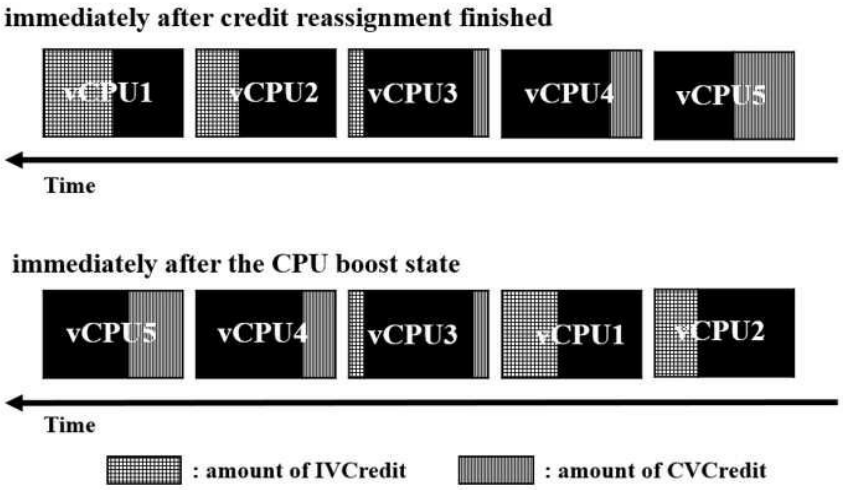
도면4


*ISACS_runqueue_info
uint64_t IOload;
int total_IOevent;
int runq_id;
s_time_t reset_time_start;
s_time_t reset_time_now;
*ISACS_eventchannel_info
boolean signal;
int IOevent_count;
int runq_id;
int dom_id;
int vCPU_id;
s_time_t event_arrival_time_start;
s_time_t event_arrival_time_now;
*ISACS_vCPU_info
uint64_t credit;
uint64_t CVCcredit;
uint64_t IVCredit;
uint64_t endcredit;
uint64_t consumed_credit;
int IOevent_count;
int runq_id;
int dom_id;
int vCPU_id;
s_time_t runq_reset_period;
boolean reset;

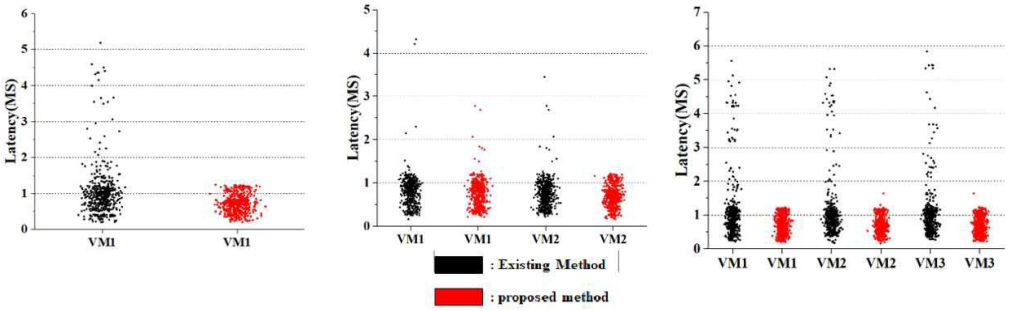
도면5


vCPU.endCredit =IVCredit
vCPU.IVCredit =IVCredit
Near the Head of Run-queue
Near the Tail of Run-queue
vCPU1
vCPU2
vCPU3
vCPU4
vCPU5
vCPU.endCredit = 0
: default credit
: Concealing Area
: IVCredit
: CVCredit

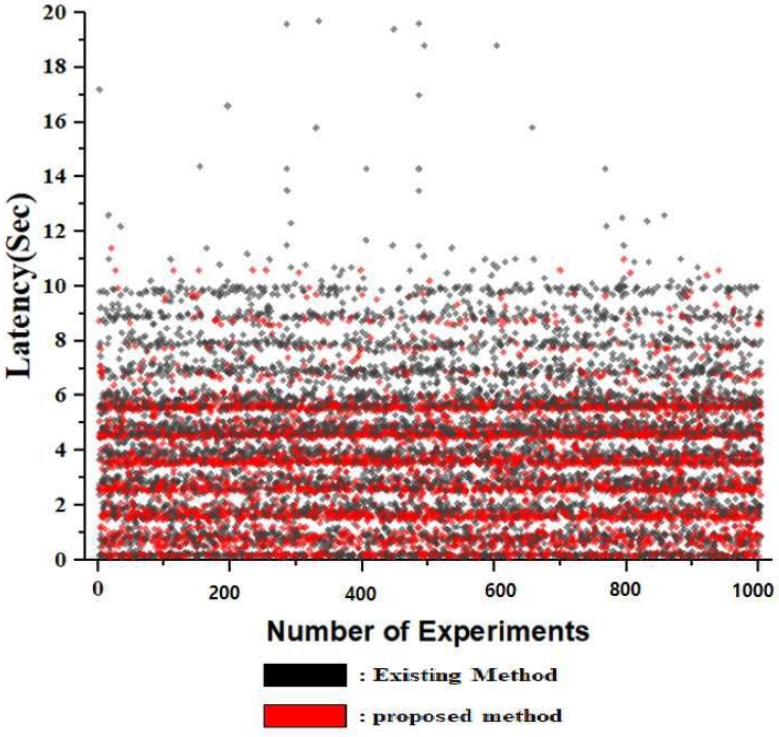
도면6


immediately after credit reassignment finished
vCPU1
vCPU2
vCPU3
vCPU4
vCPU5
Time
immediately after the CPU boost state
vCPU5
vCPU4
vCPU3
vCPU1
vCPU2
Time
: amount of IVCredit
: amount of CVCredit

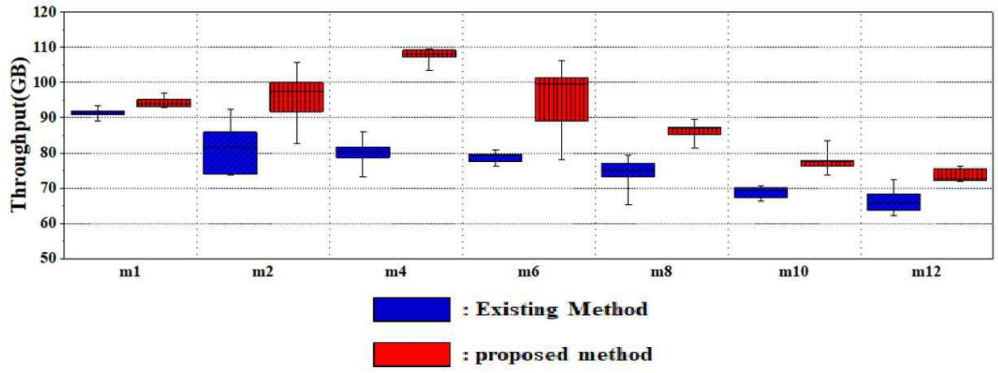
도면7


Latency(MS)
VM1
VM1
VM1
VM1
VM2
VM2
: Existing Method
: proposed method
VM1
VM1
VM2
VM2
VM3
VM3

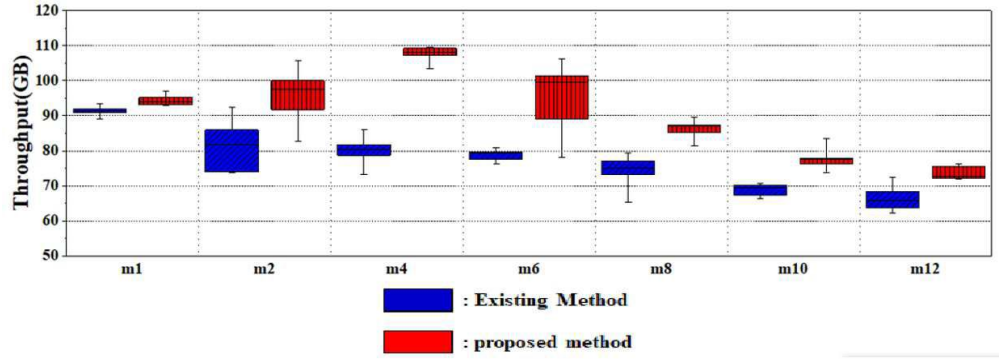
도면8


Latency(Sec)
Number of Experiments
: Existing Method
: proposed method

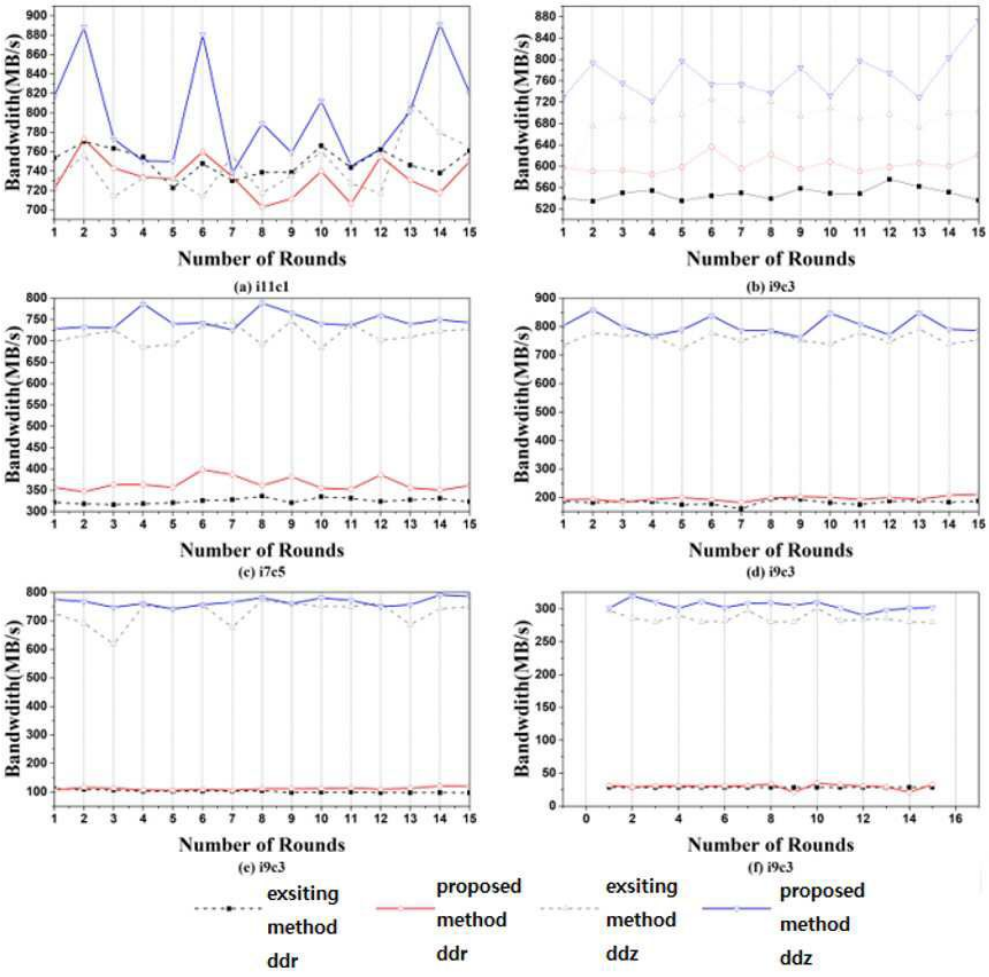
도면9


Throughput(GB)
120
110
100
90
80
70
60
50
m1
m2
m4
m6
m8
m10
m12
: Existing Method
: proposed method

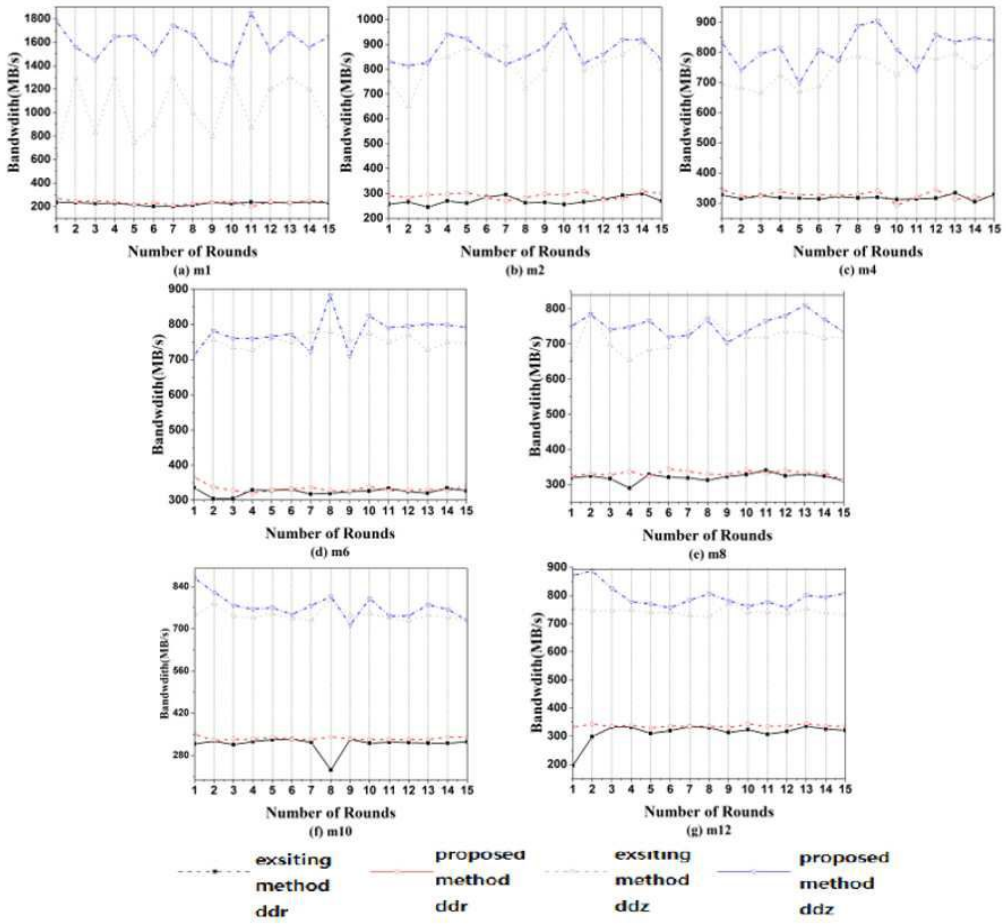
도면10


Throughput(GB)
120
110
100
90
80
70
60
50
m1
m2
m4
m6
m8
m10
m12
: Existing Method
: proposed method

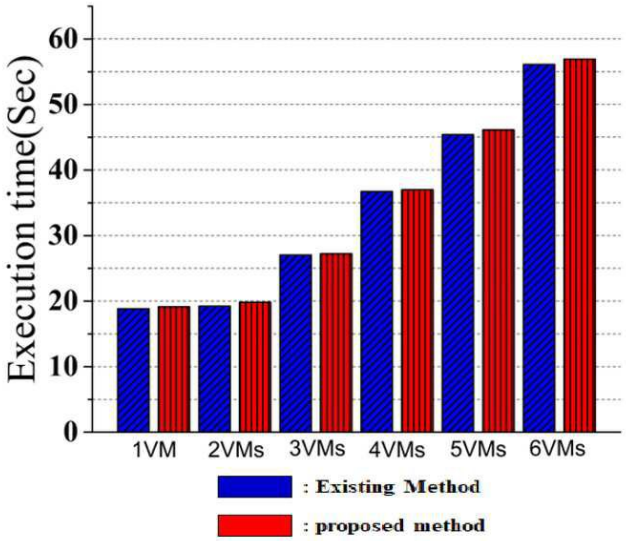
도면11


Bandwdith(MB/s)
Number of Rounds
(a) i11c1
(b) i9c3
(c) i7c5
(d) i9c3
(e) i9c3
(f) i9c3
exsiting method ddr
proposed method ddr
exsiting method ddz
proposed method ddz

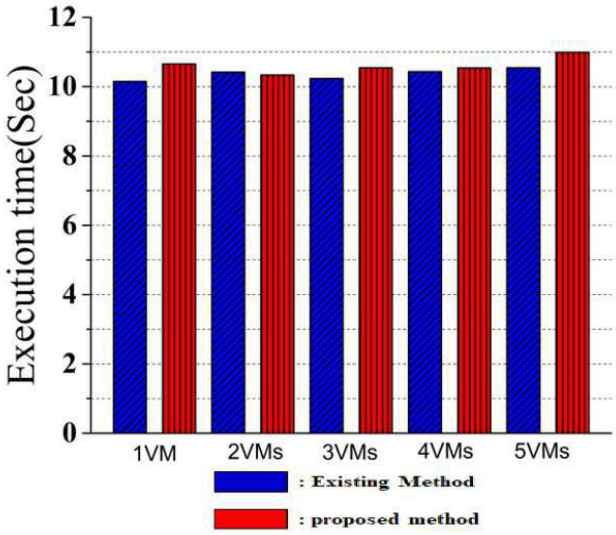
도면12


Bandwdith(MB/s)
Number of Rounds
(a) m1
(b) m2
(c) m4
(d) m6
(e) m8
(f) m10
(g) m12
exsiting method ddr
proposed method ddr
exsiting method ddz
proposed method ddz

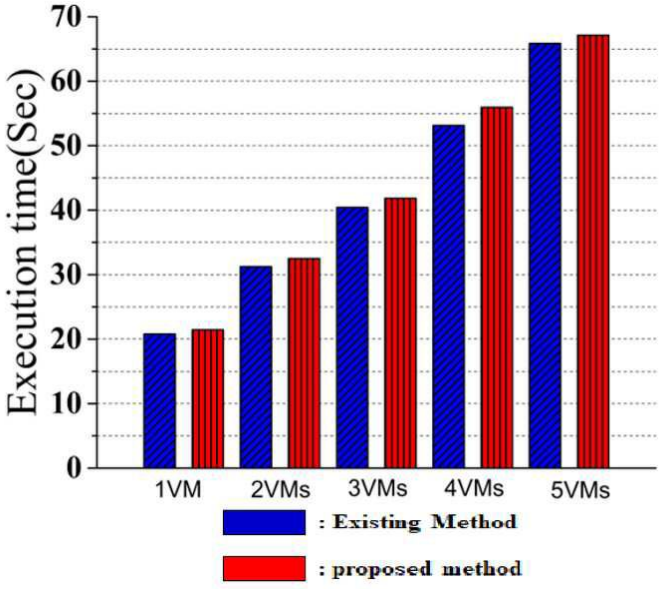
도면13


Execution time(Sec)
0
10
20
30
40
50
60
1VM
2VMs
3VMs
4VMs
5VMs
6VMs
: Existing Method
: proposed method

도면14


Execution time(Sec)
0
2
4
6
8
10
12
1VM
2VMs
3VMs
4VMs
5VMs
: Existing Method
: proposed method

도면15


Execution time(Sec)
0
10
20
30
40
50
60
70
1VM
2VMs
3VMs
4VMs
5VMs
: Existing Method
: proposed method